Caicloud

cloud. acceleration. intelligence.

才云科技

TensorFlow实现深度学习简介

郑泽宇
才云科技　首席科学家

自我介绍

竞赛金牌

优秀毕业生
十佳论文

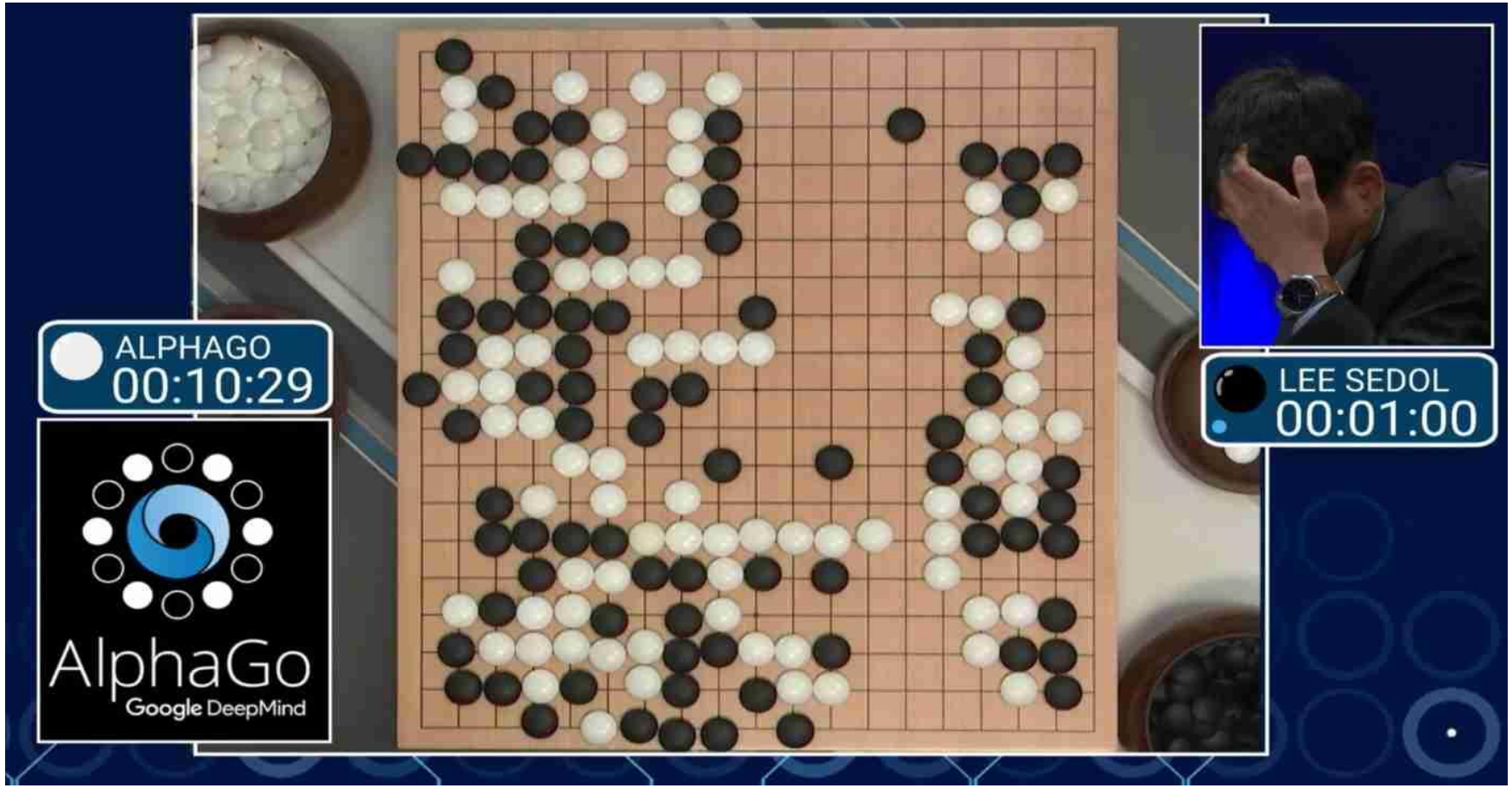
西贝尔奖学金

谷歌电商

Co-founder
首席科学家

目录

深度学习简介

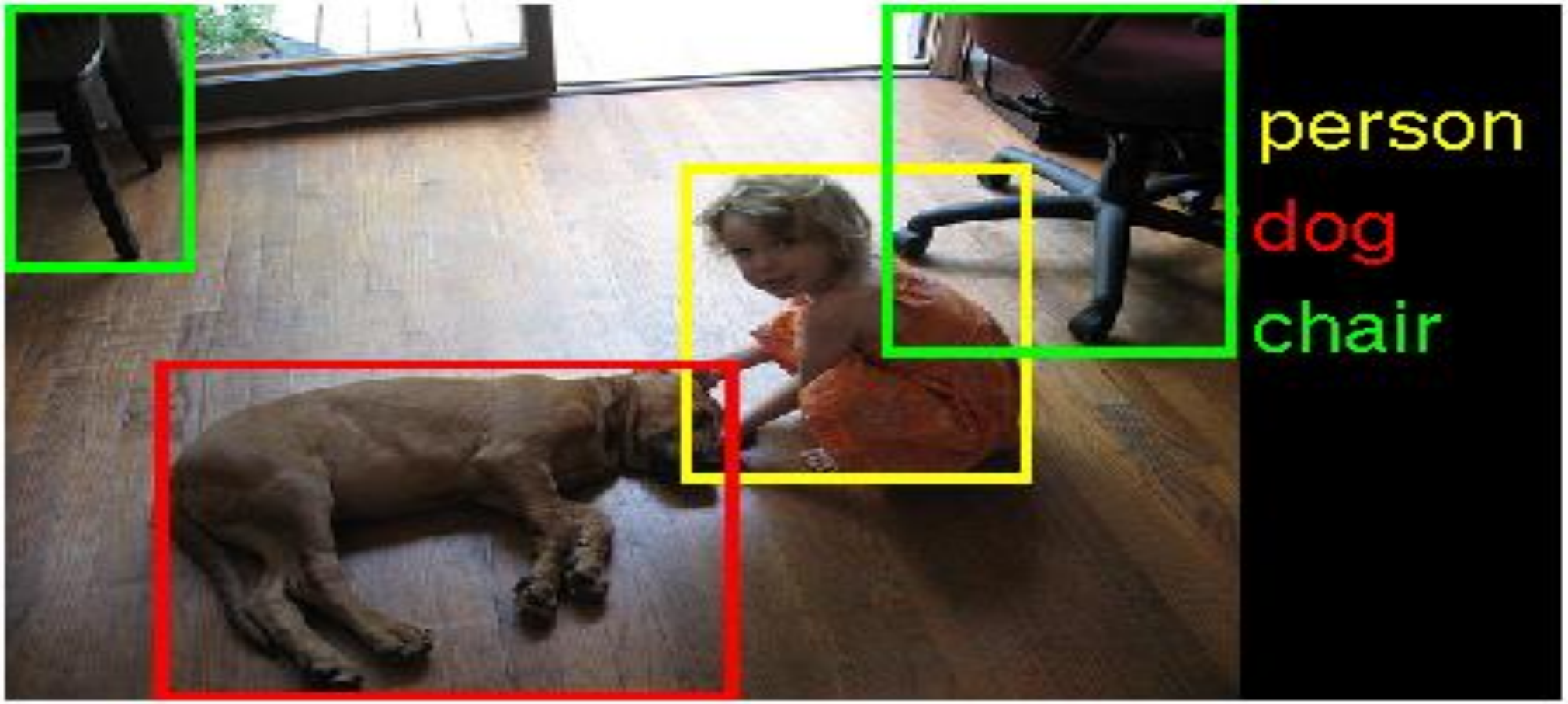


深度学习简介

深度学习简介

搜索词“deep learning”（深度学习）在Google上的热度图

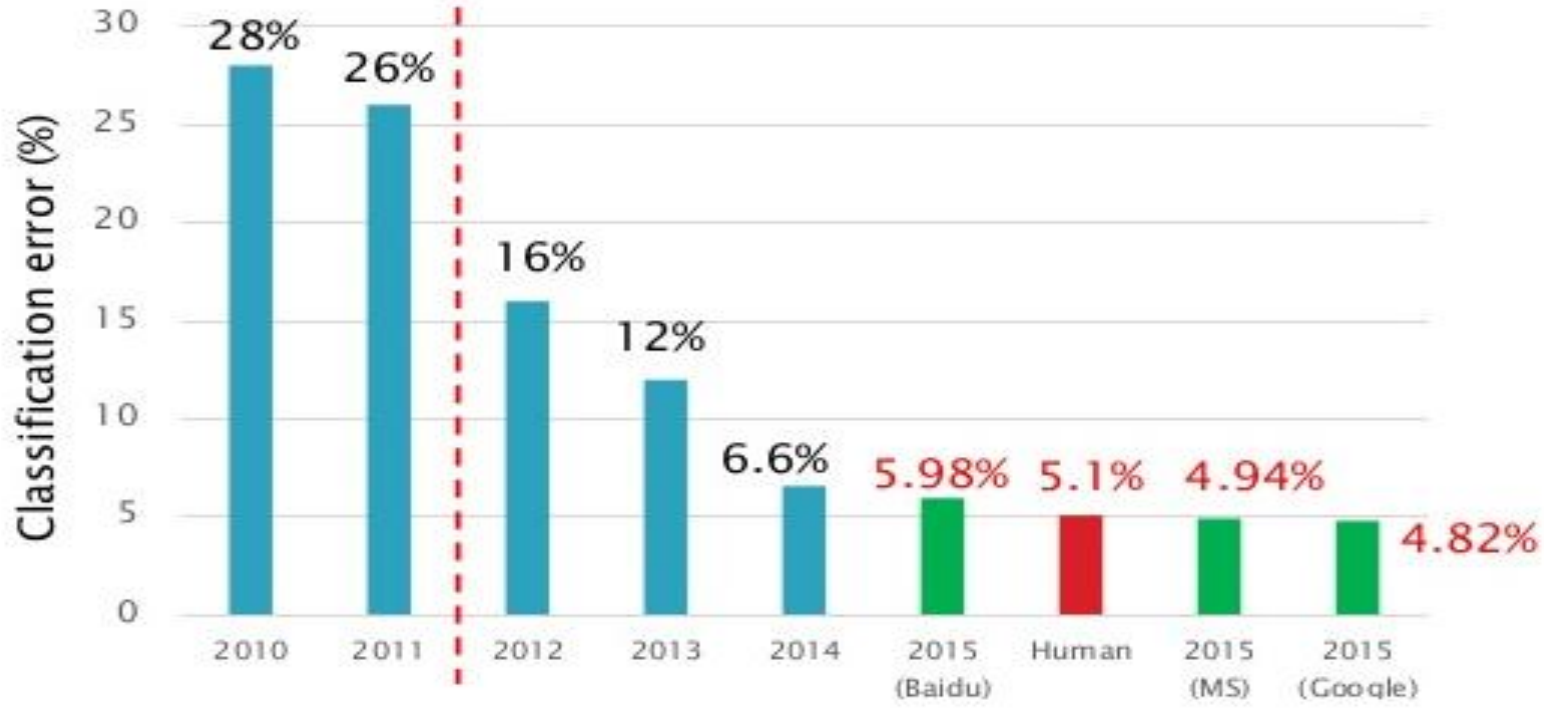
图像识别



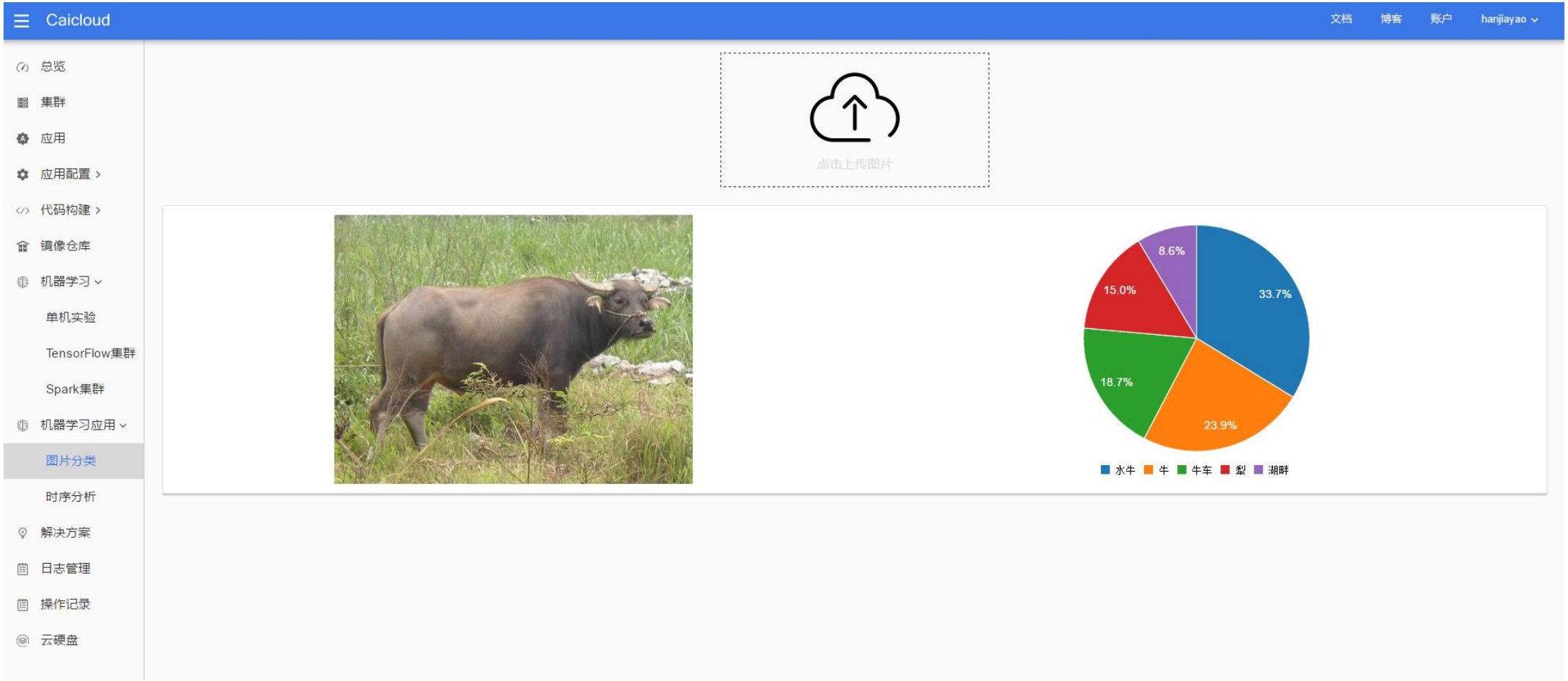
图像识别

深度学习之前

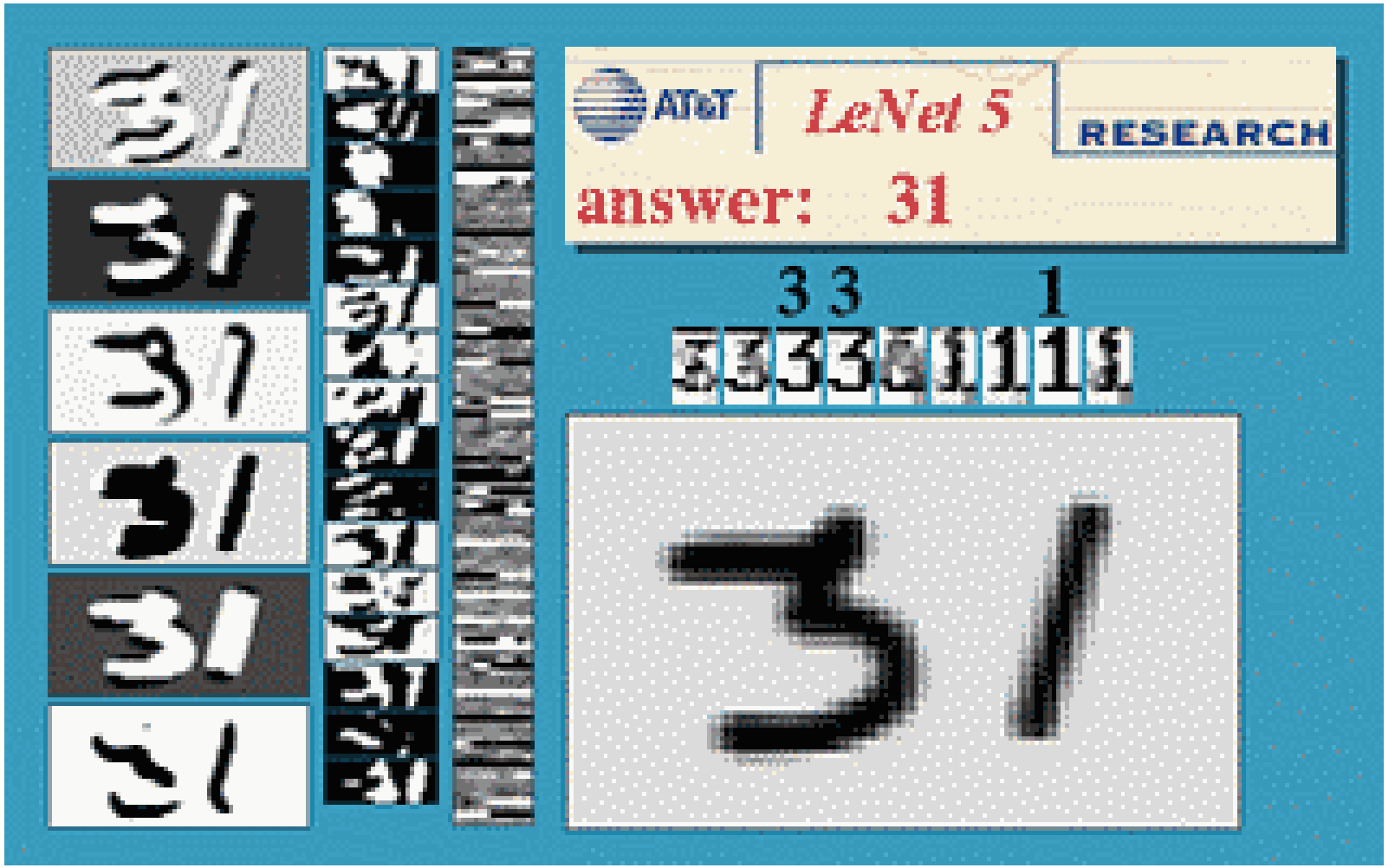
深度学习之后



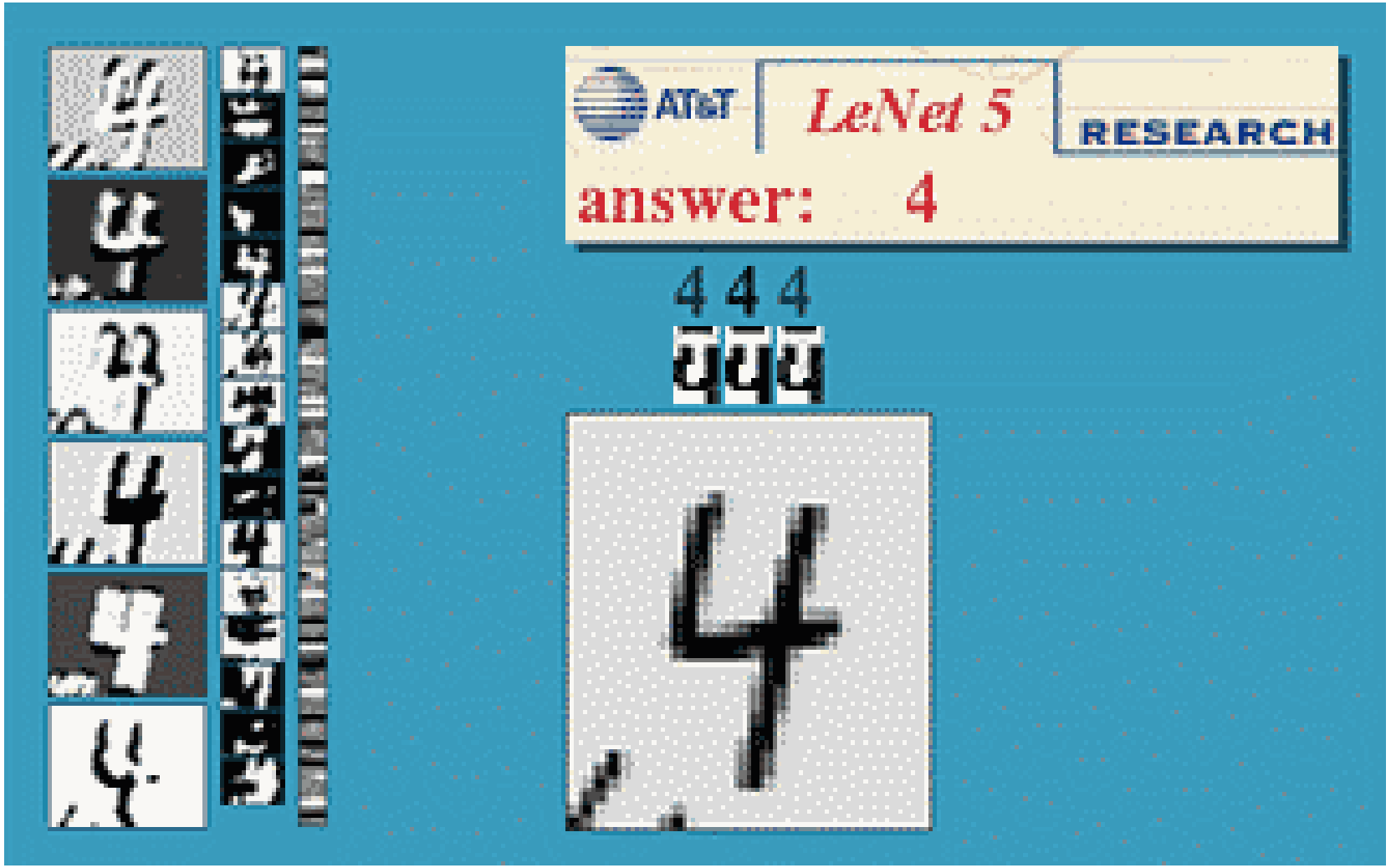
图像识别


Caicloud
文档
博客
账户
hanjiayao
总览
集群
应用
应用配置
代码构建
镜像仓库
机器学习
单机实验
TensorFlow集群
Spark集群
机器学习应用
图片分类
时序分析
解决方案
日志管理
操作记录
云硬盘
点击上传图片
8.6%
33.7%
15.0%
18.7%
23.9%
水牛
牛
牛车
梨
湖畔

图像识别



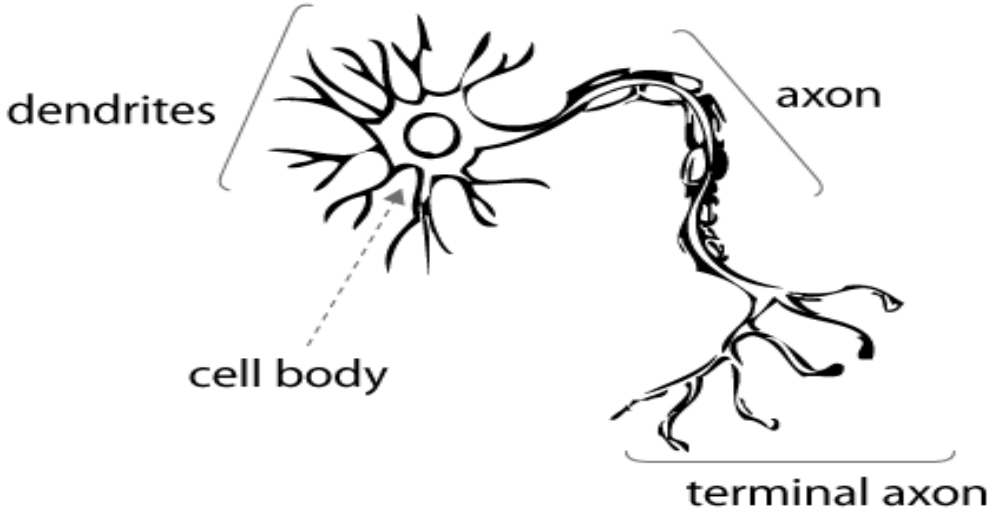
图像识别

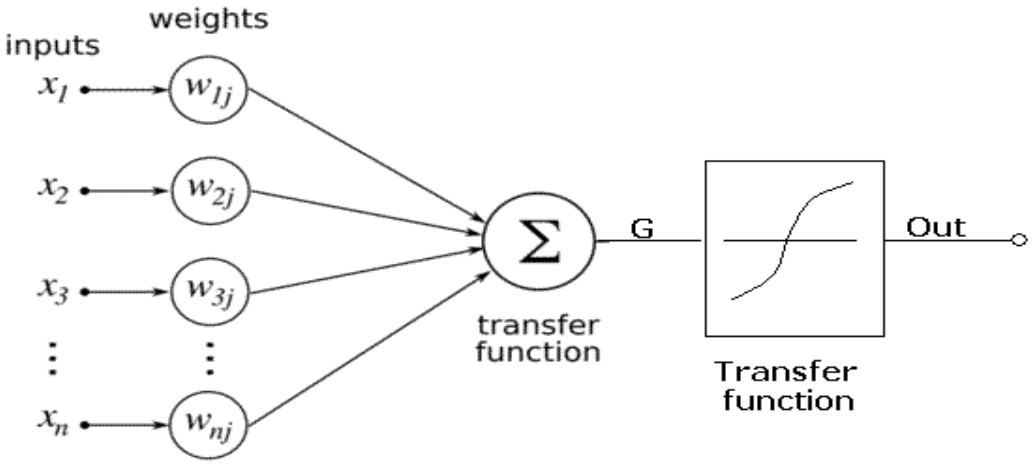


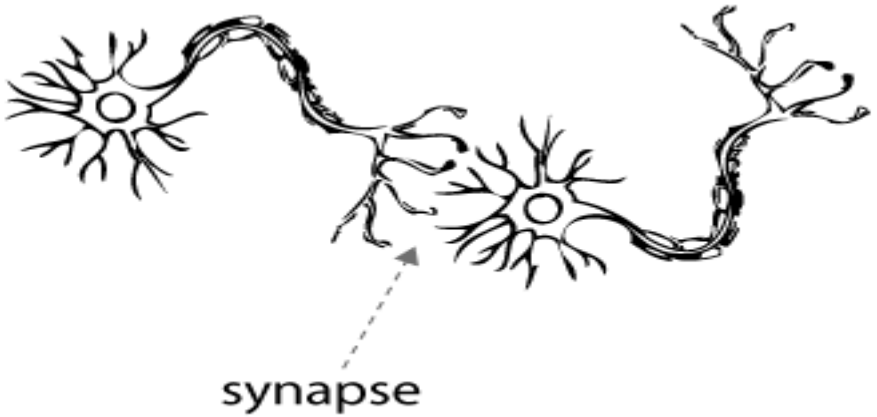
图像识别

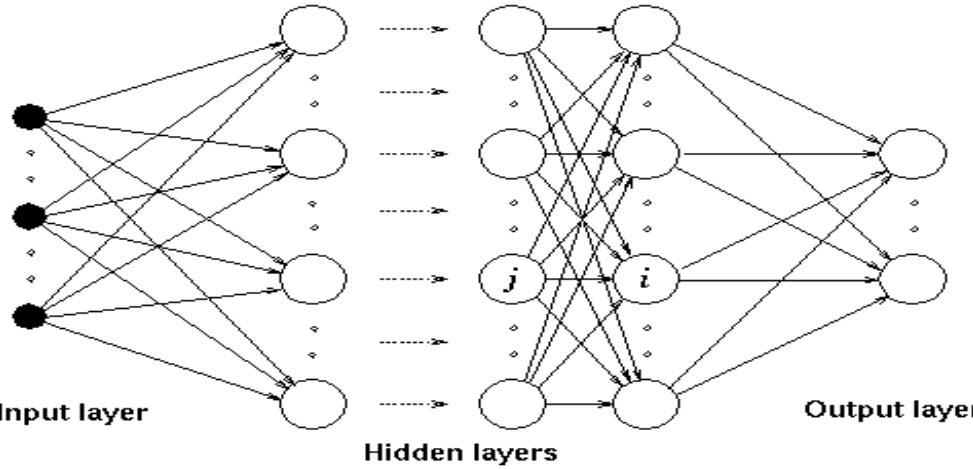
目录

神经网络模型


dendrites
axon
cell body
terminal axon

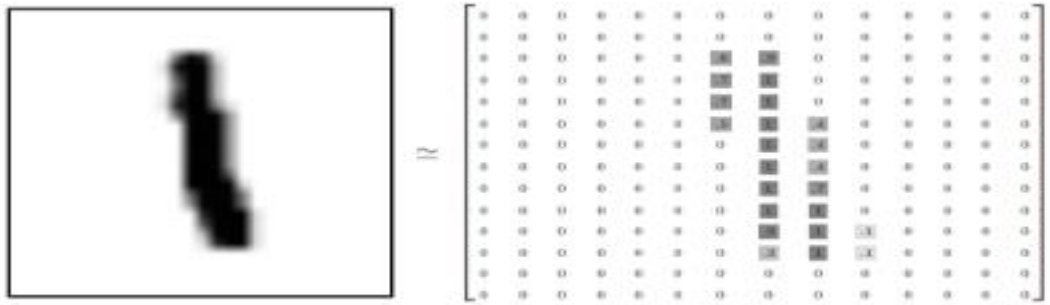

inputs
weights
x_1
x_2
x_3
x_n
w_{1j}
w_{2j}
w_{3j}
w_{nj}
Σ
G
Out
transfer function
Transfer function

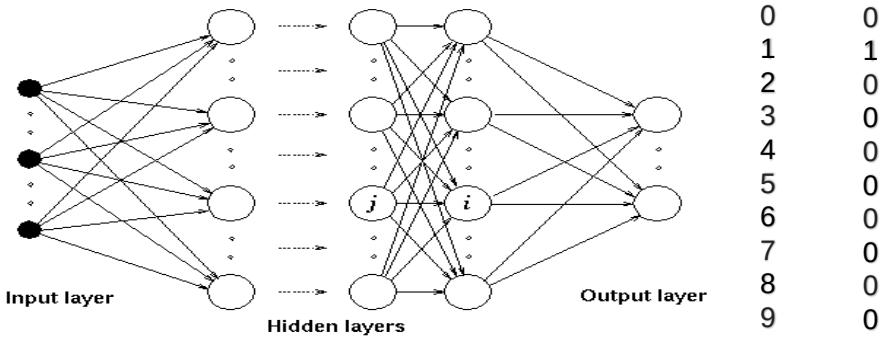

synapse


j
i
Input layer
Hidden layers
Output layer

神经网络模型

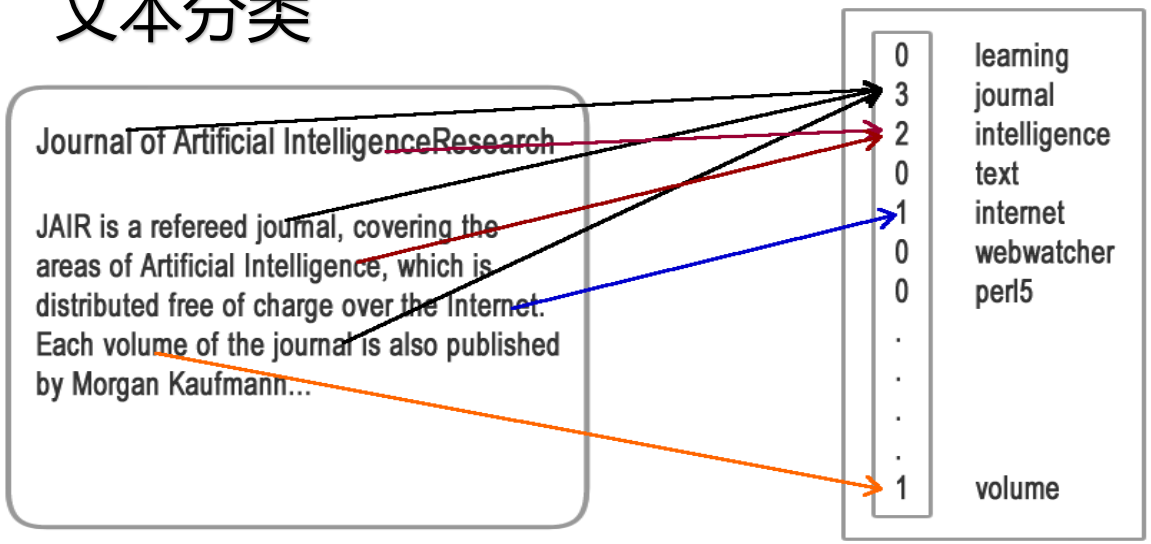
图像识别

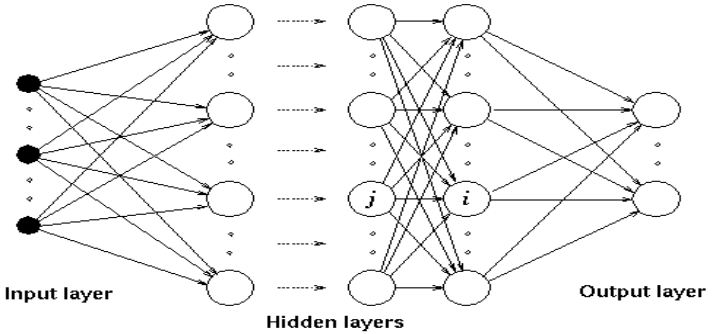




→ 1

文本分类





👎 0
👍 1

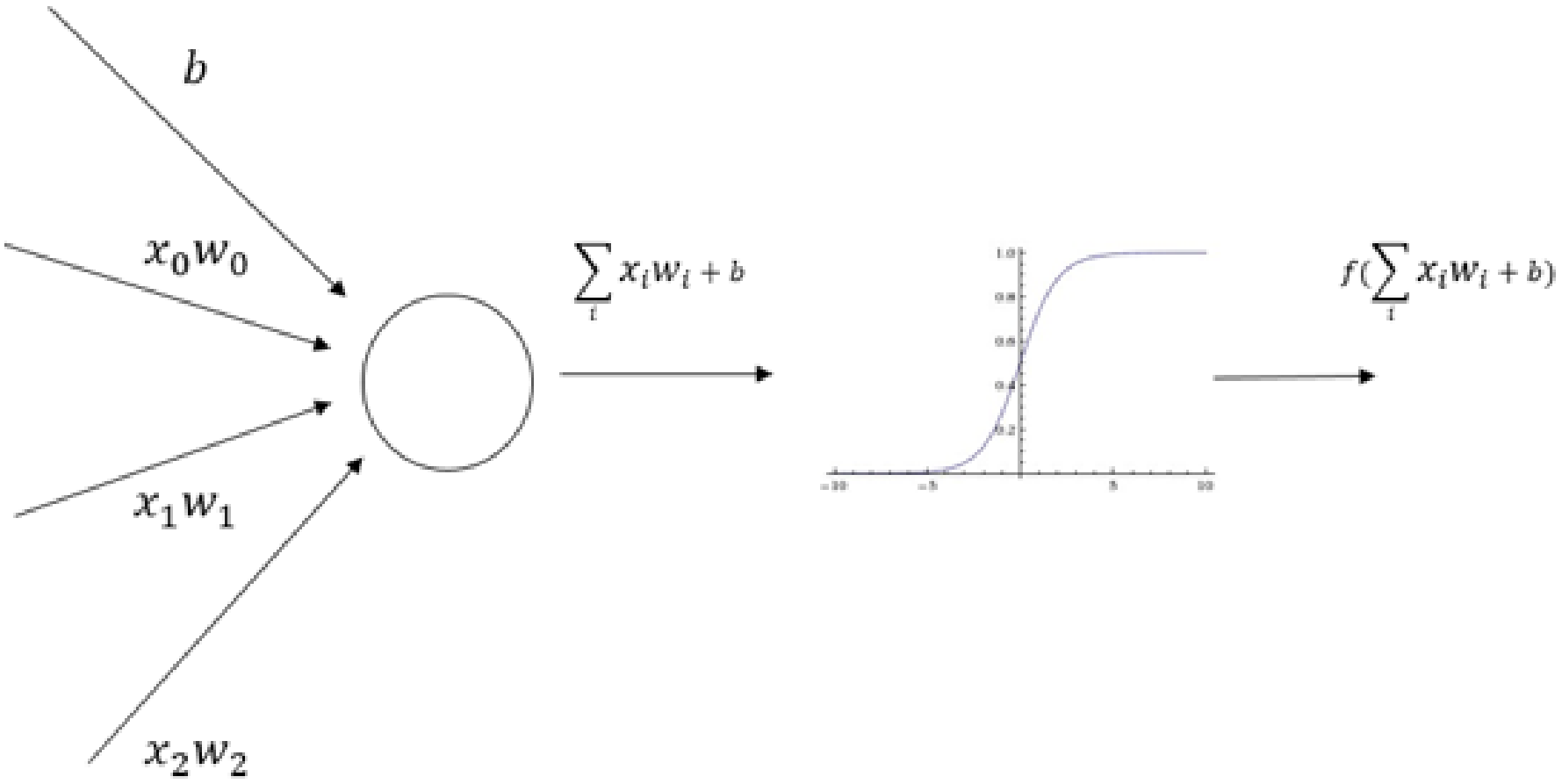
→ 🙂

监督学习和无监督学习

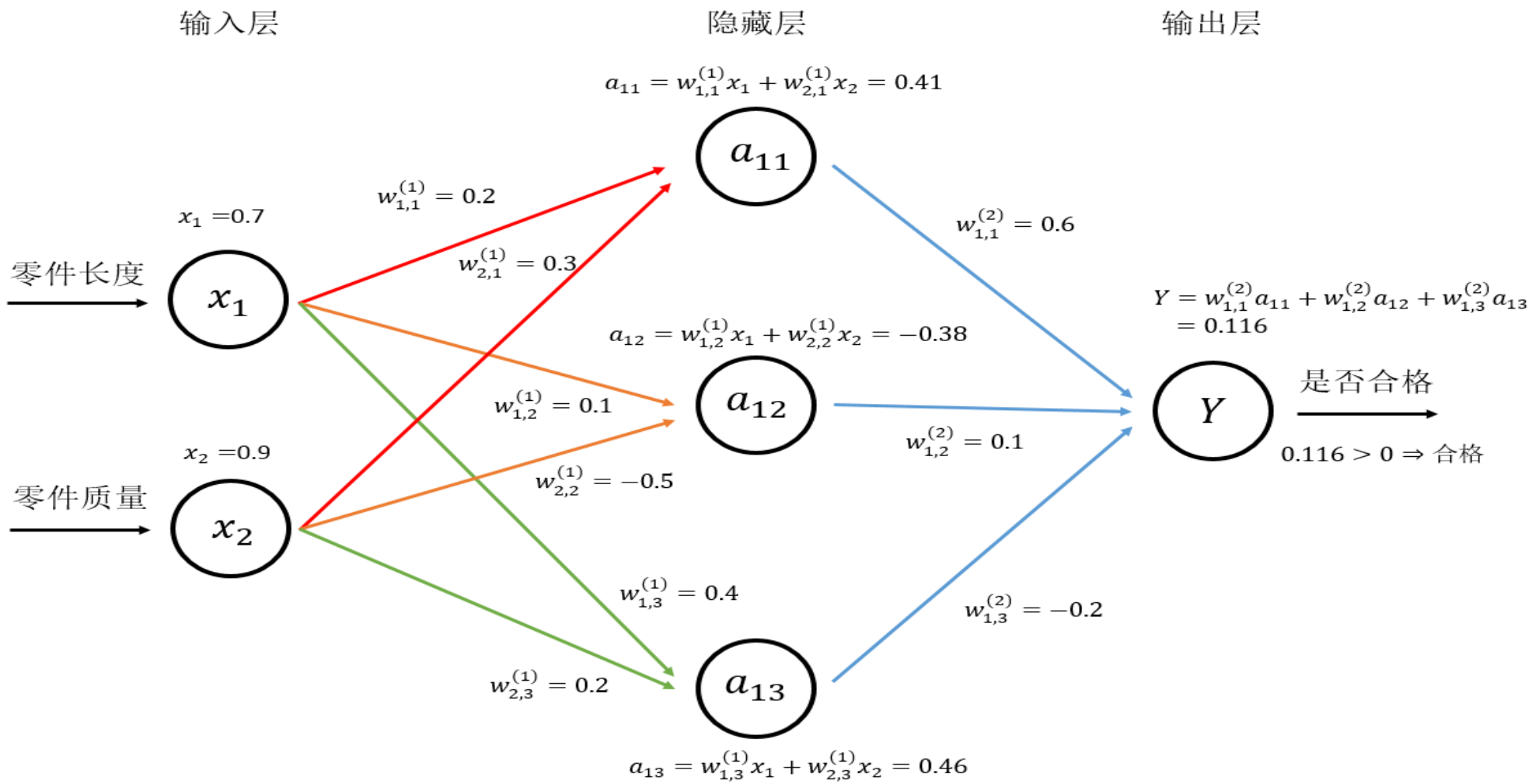
监督式学习（英语：Supervised learning），是一个机器学习中的方法，可以由训练资料中学到或建立一个模式（函数 / learning model），并依此模式推测新的实例。训练资料是由输入物件（通常是向量）和预期输出所组成。函数的输出可以是一个连续的值（称为回归分析），或是预测一个分类标签（称作分类）

非监督式学习是一种机器学习的方式，并不需要人力来输入标签。它是监督式学习和强化学习等策略之外的一种选择。典型的非监督学习有聚类等，直接从数据的特征中寻找相似性。

神经网络模型

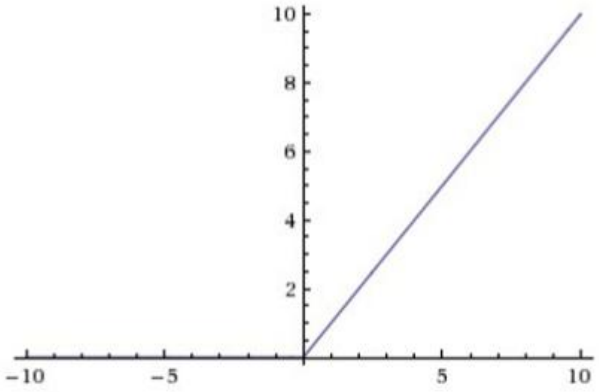


神经网络模型



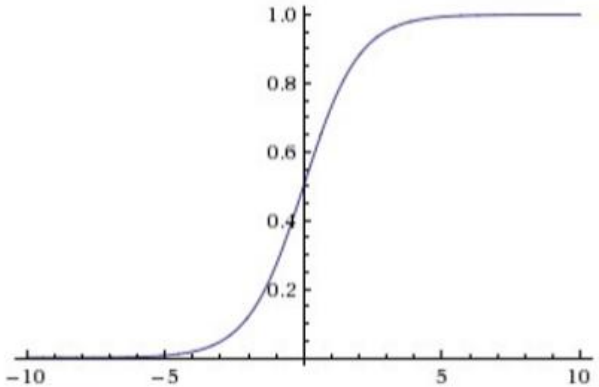
神经网络模型

几种常见的激活函数



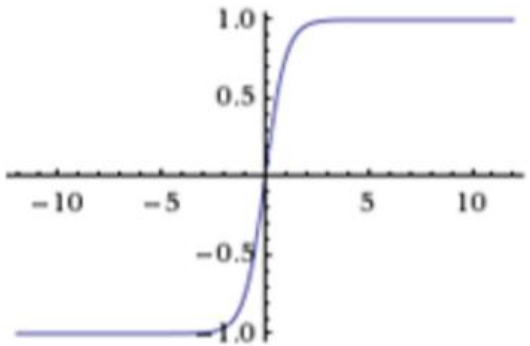
ReLU函数

$f(x) = \max(0, x)$



sigmod 函数

$f(x) = \frac{1}{1+e^{-x}}$



tanh 函数

$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$

Softmax激活层: 用于多分类问题

$$softmax(y)_i = y_i' = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}}$$

假设输出的是$[y_1, y_2, y_3 \cdots y_n]$，经过Softmax回归层后，所有的$y_i$都将被限定在[0, 1]之间， y_i的和是1。

Sigmoid激活层: 用于二分类问题

$$sigmoid(y) = \frac{1}{1+e^{-y}}$$

使用sigmoid解决二分类时一般只有一个输出y。

损失函数: 计算输出和真实标签之间的关系。

1. 交叉熵

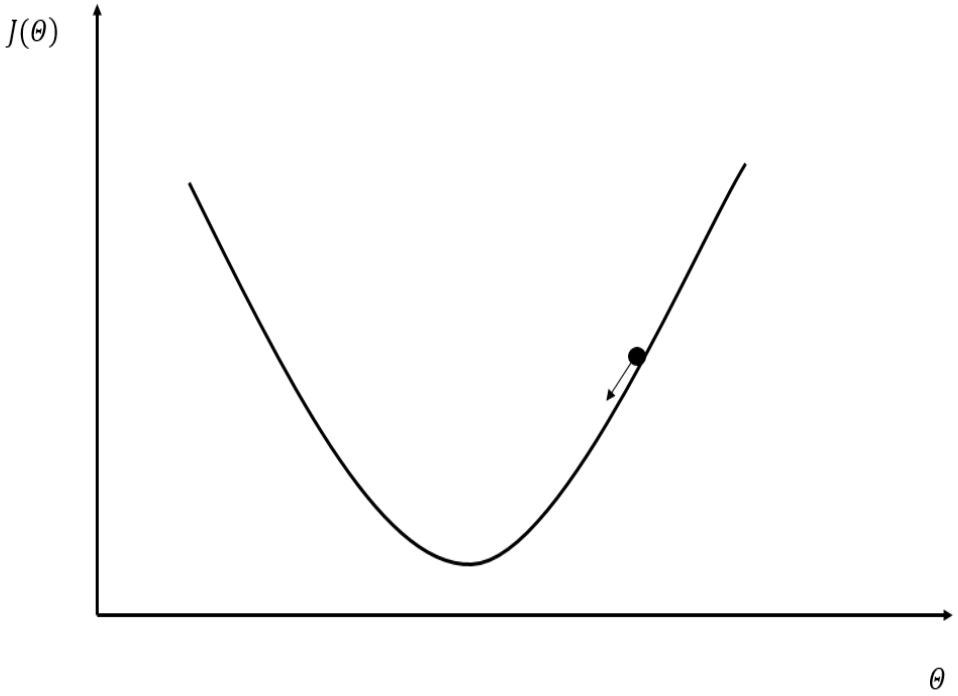
- $H(p,q) = -\sum_x p(x)\, log\, q(x)$
- p是真实的分布，q是模型预测出来的分布。交叉熵是非对称的，描述的是假设一个预测的概率分布q服从的是真实分布p所需要的平均信息量。如果预测的分布q越接近真实的分布p，那么这个信息量就越小。我们所要做的就是优化交叉熵，使其值越小，从而使得模型预测的概率分布越接近真实的概率分布。

2. MSE损失函数：即最小二乘损失函数——描述拟合函数与真实函数之间的误差。

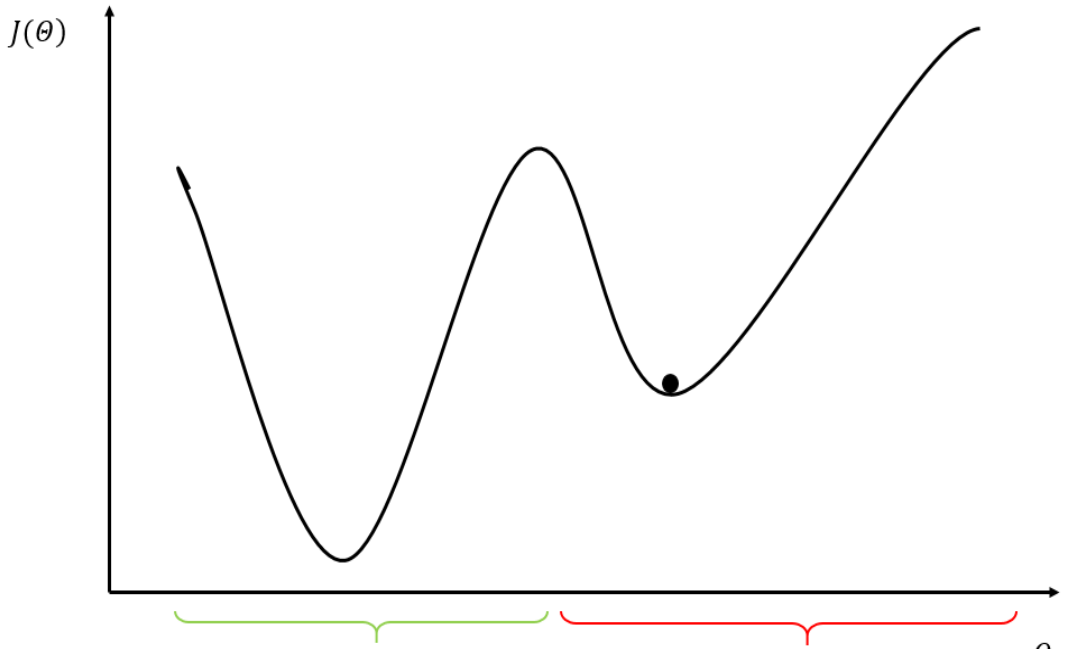
$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

3. 自定义损失函数：能够描述预测值与真实值之间的关系。

梯度下降法



最理想的情况，损失函数的
局部最小值就是全局最小值。

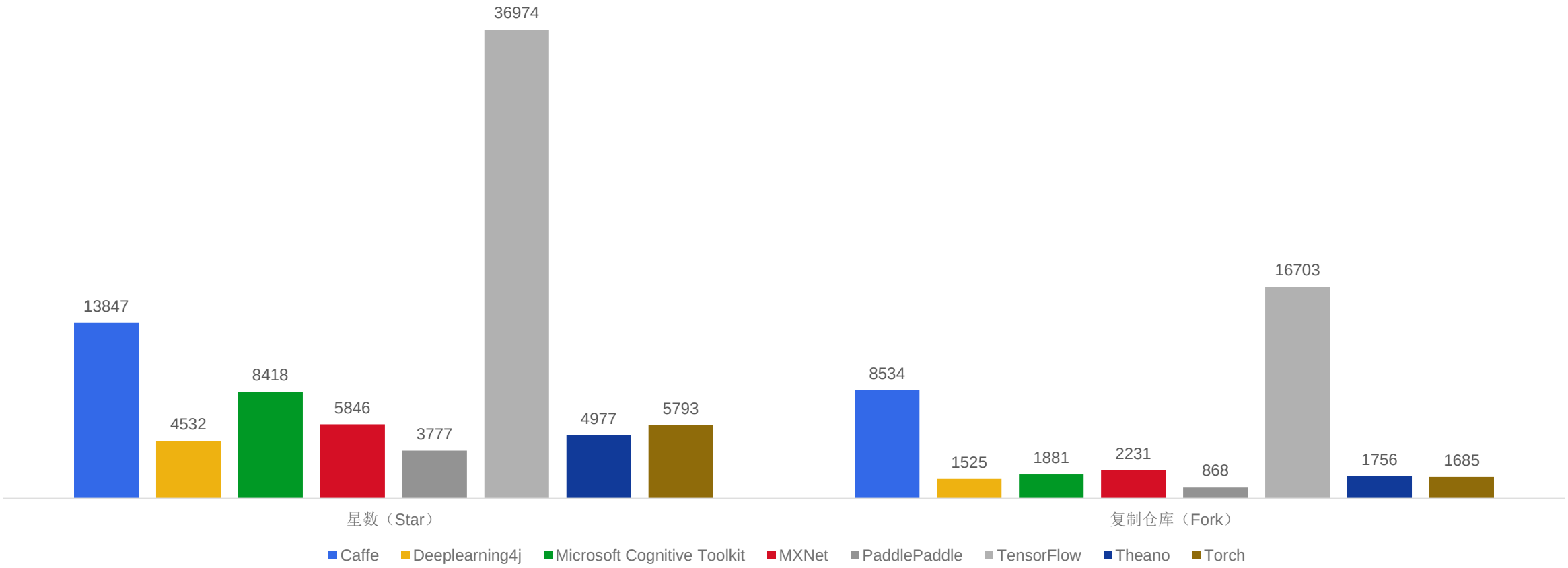


一般情况下，损失函数同时存在
局部最小值和全局最小值。这种情况下，
如果上图初始点在右边，
那么迭代会陷入局部点而不能全局收敛。

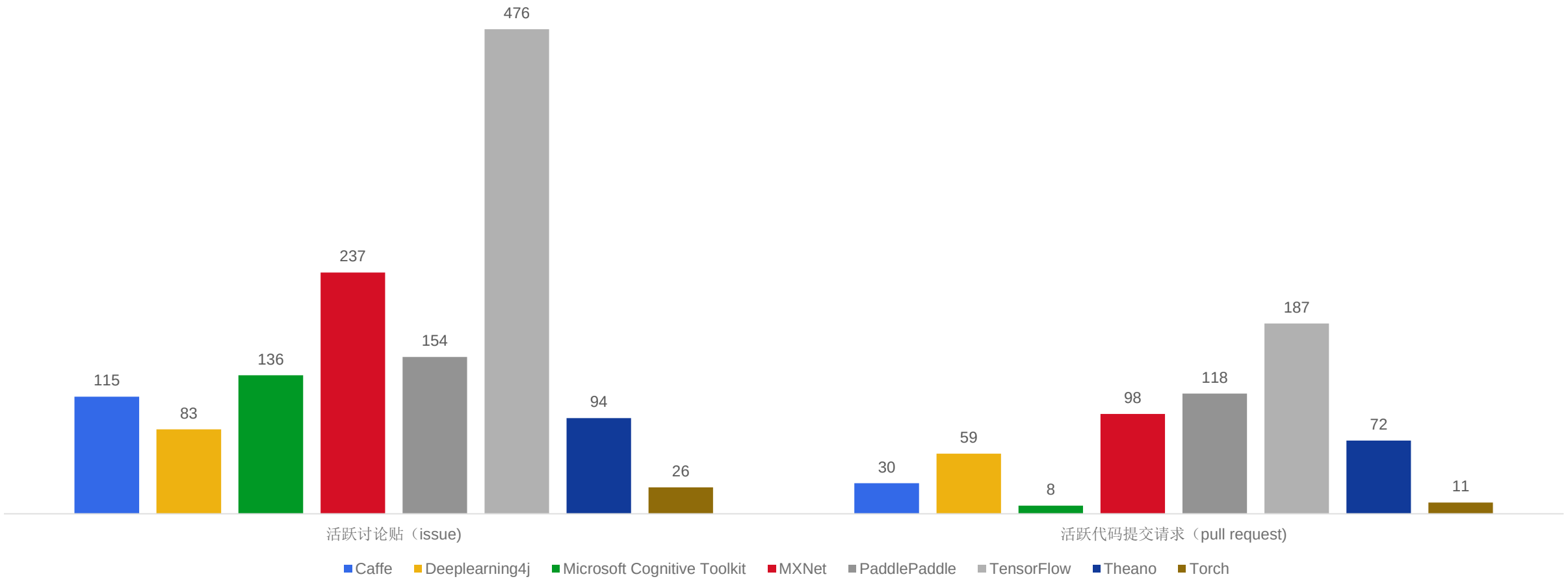
目录

TensorFlow简介

TensorFlow简介



TensorFlow简介



Tensor和Session

TensorFlow里面的变量都以tensor的形式保存，可以调用session来获取tensor的取值

```
import tensorflow as tf
a = tf.constant([1.0, 2.0], name="a")
b = tf.constant([2.0, 3.0], name="b")
result = a + b
print result
```

输出：

```
Tensor("add:0", shape=(2,), dtype=float32)
```

```
with tf.Session() as sess:
  print sess.run(result)
```

输出：

```
[3.0, 5.0]
```

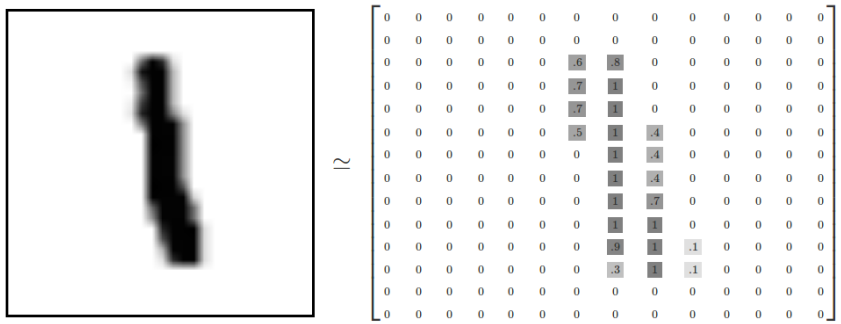
Layer的构建

```
input = tf.placeholder(tf.float32, shape=input_shape, name="input")
w = tf.Variable(tf.random_normal([input_shape, output_shape], stddev=1))
b = tf.Variable(tf.constant(0.1, shape=[output_shape]))
output = tf.nn.relu(tf.matmul(x, w) + b)
```

使用contrib.layers

```
input = tf.placeholder(tf.float32, shape=input_shape, name="input")
output = tf.contrib.layers.fully_connected(input_shape, output_shape, activation_function)
```

MNIST数据集介绍



数字图片及其像素矩阵

1. 读取数据

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("../../datasets/MNIST_data/", one_hot=True)
```

```
print "Training data size: ", mnist.train.num_examples
print "Validating data size: ", mnist.validation.num_examples
print "Testing data size: ", mnist.test.num_examples
```

```
Training data size:  55000
Validating data size:  5000
Testing data size:  10000
```

TensorFlow简介

2. 使用next_batch随机划分数据集

```
batch_size = 100
xs, ys = mnist.train.next_batch(batch_size)     # 从train的集合中选取batch_size个训练数据。
print "X shape:", xs.shape
print "Y shape:", ys.shape
```

```
X shape: (100, 784)
Y shape: (100, 10)
```

前向传播

```
def get_weight_variable(shape, regularizer):
    weights = tf.get_variable("weights", shape, initializer=tf.truncated_normal_initializer(stddev=0.1))
    if regularizer != None: tf.add_to_collection('losses', regularizer(weights))
    return weights
```

```
def inference(input_tensor, regularizer):

    #定义第一层
    with tf.variable_scope('layer1'):

        weights = get_weight_variable([INPUT_NODE, LAYER1_NODE], regularizer)
        biases = tf.get_variable("biases", [LAYER1_NODE], initializer=tf.constant_initializer(0.0))
        layer1 = tf.nn.relu(tf.matmul(input_tensor, weights) + biases)

    #定义第二层
    with tf.variable_scope('layer2'):
        weights = get_weight_variable([LAYER1_NODE, OUTPUT_NODE], regularizer)
        biases = tf.get_variable("biases", [OUTPUT_NODE], initializer=tf.constant_initializer(0.0))
        layer2 = tf.matmul(layer1, weights) + biases

    return layer2
```

TensorFlow简介

```
def train(mnist):
    # 定义输入输出placeholder。
    x = tf.placeholder(tf.float32, [None, mnist_inference.INPUT_NODE], name='x-input')
    y_ = tf.placeholder(tf.float32, [None, mnist_inference.OUTPUT_NODE], name='y-input')

    regularizer = tf.contrib.layers.l2_regularizer(REGULARIZATION_RATE)
    y = mnist_inference.inference(x, regularizer)
    global_step = tf.Variable(0, trainable=False)

    # 定义损失函数、学习率、滑动平均操作以及训练过程。
    variable_averages = tf.train.ExponentialMovingAverage(MOVING_AVERAGE_DECAY, global_step)
    variables_averages_op = variable_averages.apply(tf.trainable_variables())
    cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=y, labels=tf.argmax(y_, 1))
    cross_entropy_mean = tf.reduce_mean(cross_entropy)
    loss = cross_entropy_mean + tf.add_n(tf.get_collection('losses'))
    learning_rate = tf.train.exponential_decay(
        LEARNING_RATE_BASE,
        global_step,
        mnist.train.num_examples / BATCH_SIZE, LEARNING_RATE_DECAY,
        staircase=True)
    train_step = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss, global_step=global_step)
    with tf.control_dependencies([train_step, variables_averages_op]):
        train_op = tf.no_op(name='train')

    # 初始化TensorFlow持久化类。
    saver = tf.train.Saver()
    with tf.Session() as sess:
        tf.global_variables_initializer().run()

        for i in range(TRAINING_STEPS):
            xs, ys = mnist.train.next_batch(BATCH_SIZE)
            _, loss_value, step = sess.run([train_op, loss, global_step], feed_dict={x: xs, y_: ys})
            if i % 1000 == 0:
                print("After %d training step(s), loss on training batch is %g." % (step, loss_value))
                saver.save(sess, os.path.join(MODEL_SAVE_PATH, MODEL_NAME), global_step=global_step)
```

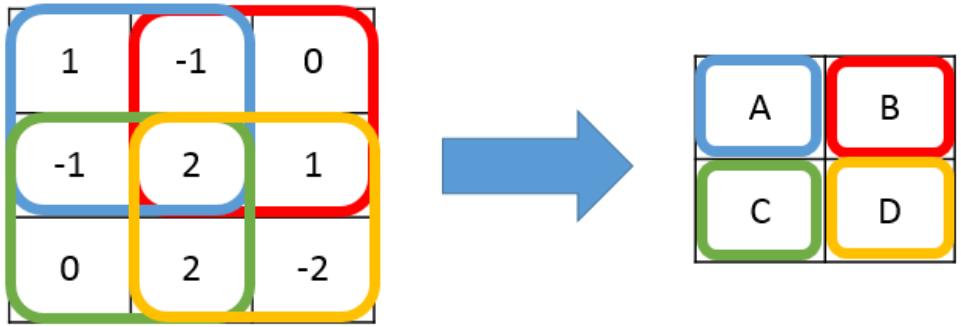
TensorFlow简介

```
def main(argv=None):
    mnist = input_data.read_data_sets("../../../datasets/MNIST_data", one_hot=True)
    train(mnist)

if __name__ == '__main__':
    main()
```

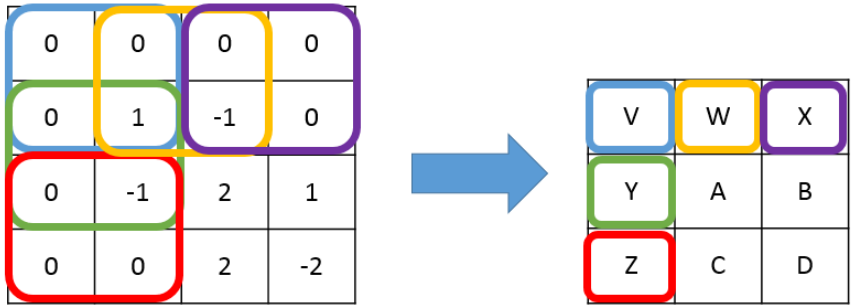
TensorFlow简介

```
# 加载的时间间隔。
EVAL_INTERVAL_SECS = 10

def evaluate(mnist):
    with tf.Graph().as_default() as g:
        x = tf.placeholder(tf.float32, [None, mnist_inference.INPUT_NODE], name='x-input')
        y_ = tf.placeholder(tf.float32, [None, mnist_inference.OUTPUT_NODE], name='y-input')
        validate_feed = {x: mnist.validation.images, y_: mnist.validation.labels}

        y = mnist_inference.inference(x, None)
        correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

        variable_averages = tf.train.ExponentialMovingAverage(mnist_train.MOVING_AVERAGE_DECAY)
        variables_to_restore = variable_averages.variables_to_restore()
        saver = tf.train.Saver(variables_to_restore)

        while True:
            with tf.Session() as sess:
                ckpt = tf.train.get_checkpoint_state(mnist_train.MODEL_SAVE_PATH)
                if ckpt and ckpt.model_checkpoint_path:
                    saver.restore(sess, ckpt.model_checkpoint_path)
                    global_step = ckpt.model_checkpoint_path.split('/')[-1].split('-')[-1]
                    accuracy_score = sess.run(accuracy, feed_dict=validate_feed)
                    print("After %s training step(s), validation accuracy = %g" % (global_step, accuracy_score))
                else:
                    print('No checkpoint file found')
                    return
            time.sleep(EVAL_INTERVAL_SECS)
```

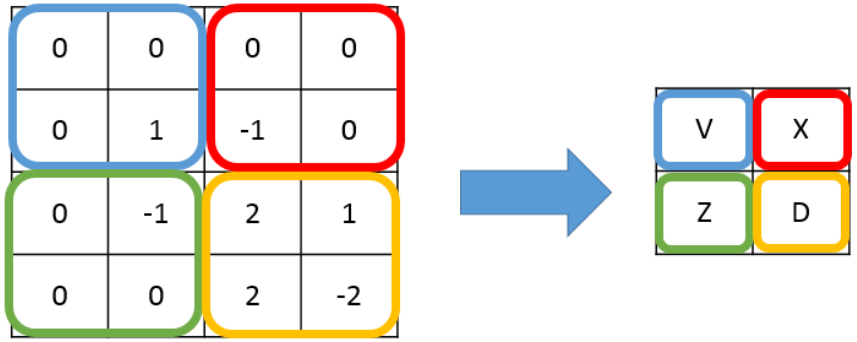
卷积神经网络



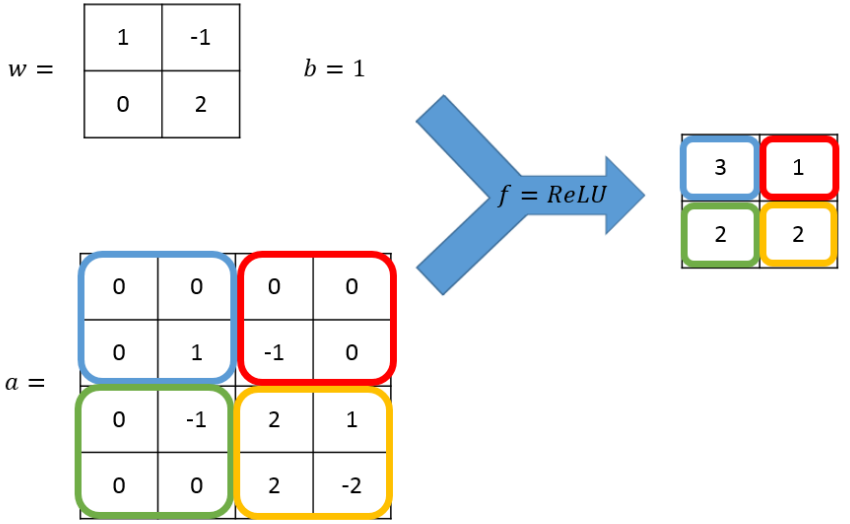
卷积层 前向传播



使用全0填充



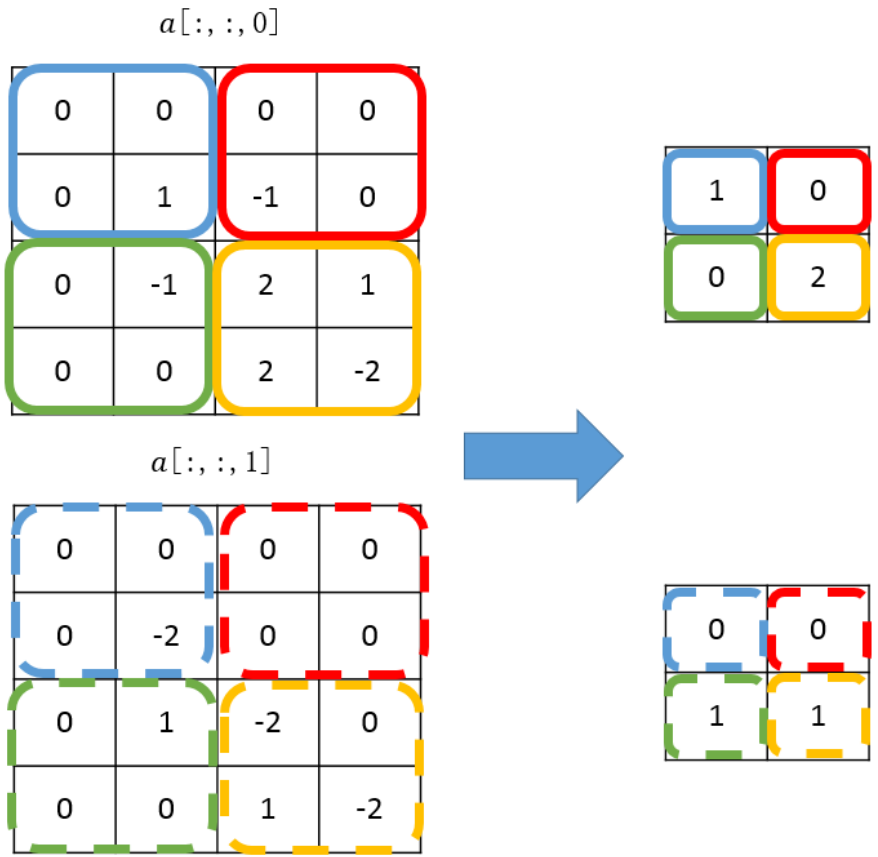
移动步长为2



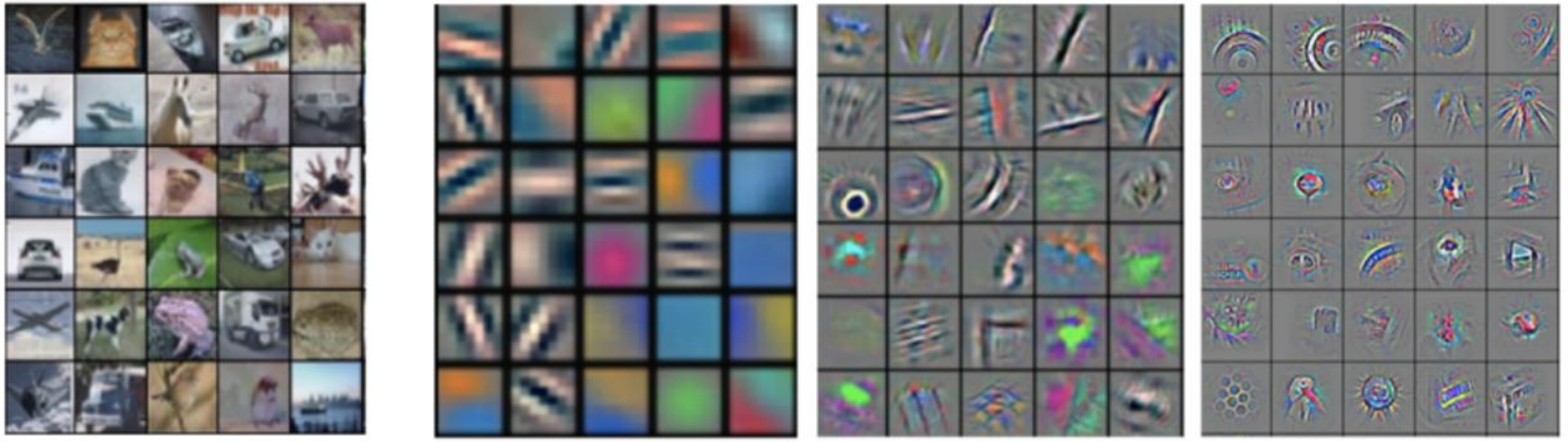
使用全0填充

池化

池化一般分为最大池化层（max pooling），
平均池化层（average pooling）

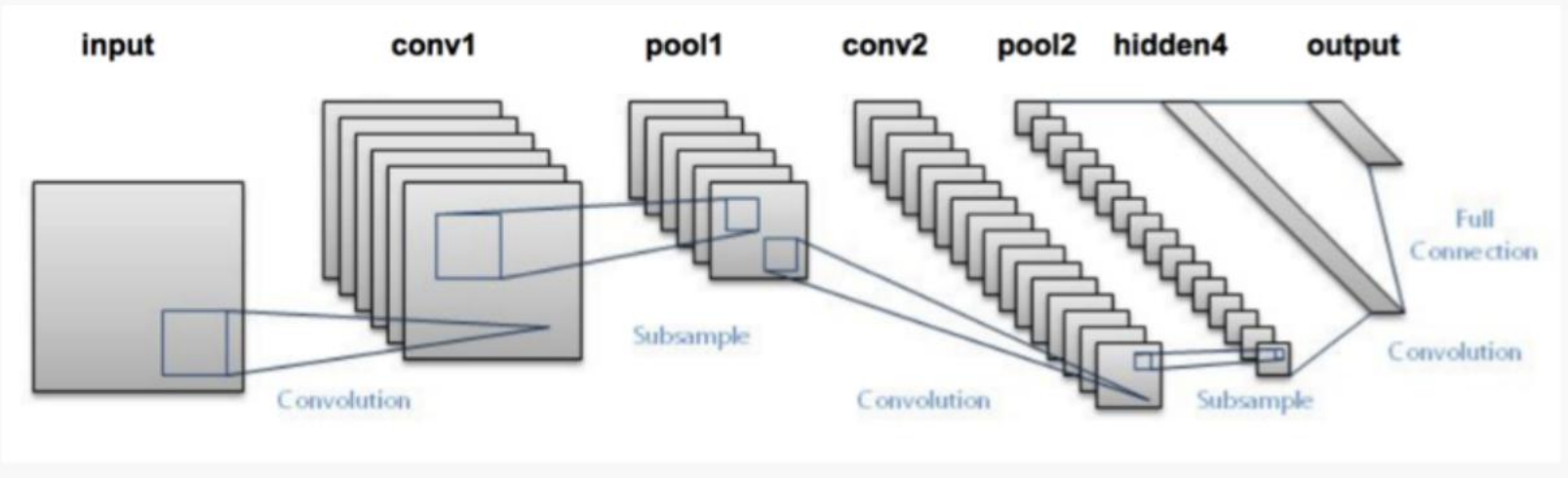


卷积相当于提取图像的特征



基础特征：图片像素　　第一层：线条　　第二层：简单形状　　第三层：复杂形状

LeNet5模型（前向传播）



input →卷积1 →池化1 →卷积2 →池化2 → 全连接1 → 全连接1 → output

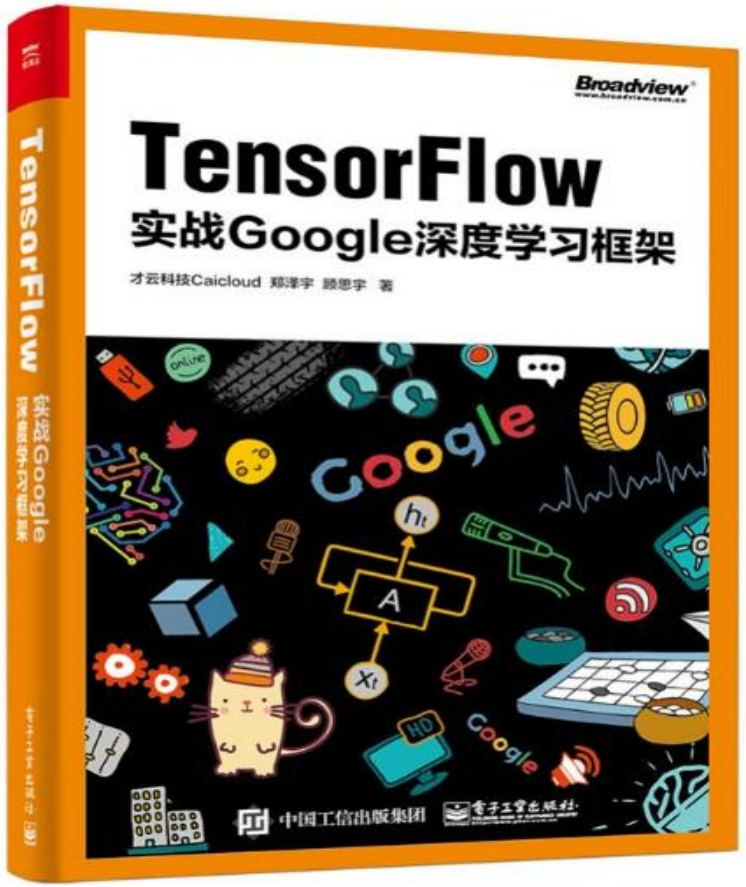
卷积神经网络

```
def inference(input_tensor, train, regularizer):
    with tf.variable_scope('layer1-conv1'):
        conv1_weights = tf.get_variable(
            "weight", [CONV1_SIZE, CONV1_SIZE, NUM_CHANNELS, CONV1_DEEP],
            initializer=tf.truncated_normal_initializer(stddev=0.1))
        conv1_biases = tf.get_variable("bias", [CONV1_DEEP], initializer=tf.constant_initializer(0.0))
        conv1 = tf.nn.conv2d(input_tensor, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
        relu1 = tf.nn.relu(tf.nn.bias_add(conv1, conv1_biases))

    with tf.name_scope("layer2-pool1"):
        pool1 = tf.nn.max_pool(relu1, ksize = [1,2,2,1],strides=[1,2,2,1],padding="SAME")

    with tf.variable_scope("layer3-conv2"):
        conv2_weights = tf.get_variable(
            "weight", [CONV2_SIZE, CONV2_SIZE, CONV1_DEEP, CONV2_DEEP],
            initializer=tf.truncated_normal_initializer(stddev=0.1))
        conv2_biases = tf.get_variable("bias", [CONV2_DEEP], initializer=tf.constant_initializer(0.0))
        conv2 = tf.nn.conv2d(pool1, conv2_weights, strides=[1, 1, 1, 1], padding='SAME')
        relu2 = tf.nn.relu(tf.nn.bias_add(conv2, conv2_biases))

    with tf.name_scope("layer4-pool2"):
        pool2 = tf.nn.max_pool(relu2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
        pool_shape = pool2.get_shape().as_list()
        nodes = pool_shape[1] * pool_shape[2] * pool_shape[3]
        reshaped = tf.reshape(pool2, [pool_shape[0], nodes])
```

卷积神经网络

```
with tf.variable_scope('layer5-fc1'):
    fc1_weights = tf.get_variable("weight", [nodes, FC_SIZE],
                                  initializer=tf.truncated_normal_initializer(stddev=0.1))
    if regularizer != None: tf.add_to_collection('losses', regularizer(fc1_weights))
    fc1_biases = tf.get_variable("bias", [FC_SIZE], initializer=tf.constant_initializer(0.1))

    fc1 = tf.nn.relu(tf.matmul(reshaped, fc1_weights) + fc1_biases)
    if train: fc1 = tf.nn.dropout(fc1, 0.5)

with tf.variable_scope('layer6-fc2'):
    fc2_weights = tf.get_variable("weight", [FC_SIZE, NUM_LABELS],
                                  initializer=tf.truncated_normal_initializer(stddev=0.1))
    if regularizer != None: tf.add_to_collection('losses', regularizer(fc2_weights))
    fc2_biases = tf.get_variable("bias", [NUM_LABELS], initializer=tf.constant_initializer(0.1))
    logit = tf.matmul(fc1, fc2_weights) + fc2_biases

return logit
```

TensorFlow简介



https://item.jd.com/12125572.html?dist=jd

https://github.com/caicloud/tensorflow-tutorial

Caicloud

cloud. acceleration. intelligence.

才云科技

谢谢大家！